[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/350,492	07/09/1999	VENKATESH KRISHNAN	10981455-1	8205

22879 7590 01/20/2004

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
TANG, KENNETH	
ART UNIT	**PAPER NUMBER**
2127	14

DATE MAILED: 01/20/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

[illegible]

PTO-90C (Rev. 10/03)

Office Action Summary	**Application No.** 09/350,492	**Applicant(s)** KRISHNAN ET AL.	
	Examiner Kenneth Tang	**Art Unit** 2127	

-- The MAILING DATE of this communication appears on th cover sheet with th correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 17 November 2003.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 30-52 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 30-52 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

13)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____.
4) ☐ Interview Summary (PTO-413) Paper No(s). _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. This final action is in response to paper number 13, Amendment/Response C, filed on 6/5/03.

2. Applicant's arguments have been fully considered but are now moot due to the new grounds of rejections. Claims 30-52 are presented for examination.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. **Claims 30-35 and 47-52 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nelson (US 2001/0049686) in view of Poole (US 6,314,445 B1).**

4. As to claims 30 and 47, Nelson teaches a system for adapting threads support in a virtual machine to an underlying platform of the virtual machine, comprising:

- threads interface layer that provides a standard threads interface for a set of threads associated with an application program of the virtual machine such that the standard threads interface does not depend on the underlying platform *("Java Virtual Machine", [0039], "In particular, the combination of Java programming code and the Java Virtual*

Machine ("JVM") allows the creation of platform-independent software that can be installed dynamically and/or remotely", [0006]);

- native threads interface layer for adapting the threads interface layer to the underlying platform such that a set of routines in the threads interface layer use a set of routines in the native threads interface layer to support the threads *("Java Native Interface layer, [0039], Fig. 6, and threads, [0039], "Platform layer", [0032]-[0033], "translate computer instructions encoded in a first computer instruction language into a Java native interface ("JNI") computer instruction language", [0010]).*

Nelson teaches a Java Virtual Machine, which has a standard threads interface for a set of threads interacting with a Java native interface with platform independence *(see above)* but fails to explicitly teach the threads are associated with an application program of the virtual machine such that the standard threads interface does not depend on the underlying platform. However, Poole teaches a virtual machine application (and interaction with a native threads interface layer, JNI) associated with a threads interface without depending on the underlying platform *("One reason for its success is that it incorporates a virtual machine layer as an interface between the AS/400 computer and AS/400 applications. This allows the platform to be defined by the virtual machine software rather than the platform hardware and the advantage follows that existing software need not be rewritten for new hardware.", col. 1, lines 15-20, "The idea of platform independence is not exclusive to the IBM AS/400 and has also been developed in another area. In particular the Java programming language was originally designed for use in the consumer electronics industry so that microprocessors in kettles, videos and cameras could be programmed in the same language. This platform independent property in the Java language* *was found to be ideal for programs distributed on the internet where different platforms are connected together in a network.* *Applications* *are written in high level Java source code, compiled into Java byte code by a Java compiler and interpreted by the Java* *Virtual Machine* *(JVM). Its main advantage is that*

it is 'write once run anywhere', that is, *platform independent* *and portable.", col. 1, lines 40-51, "IBM's AS/400 and Sun Microsystem's Java therefore have a* *virtual machine interface* *in common and when used together, a Java* *application* *running on an AS/400 uses a JVM to* *interface* *to the AS/400* *virtual machine which interfaces* *to the AS/400 computer. This combination is advantageous in that it allows Java applications to run on present and future AS/400 computers. However a problem exists when the JVM needs to access the stack pointer of the AS/400 computer: since there is no facility in the AS/400 virtual machine to do this,*

there can be no direct access from the JVM.", col. 1, lines 55-65, "A native call is a procedure call to code external to the Java environment, for example to the platform operating system OS/400 or another platform specific routine. The Java specification states that native calls should be allowed although if implemented the Java application containing it will not be 100% Java and portable to other platforms. Native calls are necessary so that existing legacy systems may be integrated with new Java applications, *for example a Java interface to an existing database application**.", col. 2, lines 8-15, "The JVM 10A comprises a native function handling component, the Java Native* *Interface* *(JNI) 32A", col. 4, lines 42-44).* It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of having threads that are associated with an application program of the virtual machine with platform independence for the reason of increasing convenience *(This* *platform independent* *property in the Java language was found to be ideal for programs distributed on the internet where different platforms are connected together in a network. Applications are written in high level Java source code, compiled into Java byte code by a Java compiler and interpreted by the Java Virtual Machine (JVM). Its main advantage is thatit is 'write once run anywhere', that is, platform independent and portable.", col. 1, lines 40-51).*

5. As to claim 31, Nelson teaches the system of claim 30, wherein the native threads interface layer is adapted to an operating system of the underlying platform (*Java Native Interface ("JNI") Layer 604 includes a library of C and/or C++ methods configured to define a Java Virtual Machine ("JVM") that provides translation of CMIS to Java.", "this layer also includes Solaris threads configured to provide additional support when the MPA is running in conjunction with the Solaris operating system", [0039]).*

6. As to claim 32, Nelson teaches the system of claim 31, wherein the native threads interface layer is adapted to use a set of thread support routines provided by the operating system (*Java Native Interface ("JNI") Layer 604 includes a library of C and/or C++ methods configured to define a Java Virtual Machine ("JVM") that provides translation of CMIS to Java.", "this layer also includes Solaris threads configured to provide additional support when the MPA is running in conjunction with the Solaris operating system", [0039])..*

7. As to claim 33, Nelson teaches the system of claim 31, wherein the native threads interface layer is adapted to use a set of routines provided by the operating system that perform

equivalent functions of functions in the native threads interface layer (*Java Native Interface ("JNI") Layer 604 includes a library of C and/or C++ methods configured to define a Java Virtual Machine ("JVM") that provides translation of CMIS to Java.", "this layer also includes Solaris threads configured to provide additional support when the MPA is running in conjunction with the Solaris operating system", [0039]).*

8. As to claim 34, Nelson teaches the system of claim 30, wherein the native threads interface layer is adapted to a hardware architecture of the underlying platform (*Java Native Interface ("JNI") Layer 604 includes a library of C and/or C++ methods configured to define a Java Virtual Machine ("JVM") that provides translation of CMIS to Java.", "this layer also includes Solaris threads configured to provide additional support when the MPA is running in conjunction with the Solaris operating system", [0039]).*

9. As to claim 35, Nelson teaches the system of claim 30, wherein the standard threads interface is a Java threads class (*Java Native Interface ("JNI") Layer 604 includes a library of C and/or C++ methods configured to define a Java Virtual Machine ("JVM") that provides translation of CMIS to Java.", "this layer also includes Solaris threads configured to provide additional support when the MPA is running in conjunction with the Solaris operating system", [0039]).*

10. As to claim 48, it is rejected for the same reasons as stated in the rejection of claim 31.

11. As to claim 49, it is rejected for the same reasons as stated in the rejection of claim 32.

12. As to claim 50, it is rejected for the same reasons as stated in the rejection of claim 33.

13. As to claim 51, it is rejected for the same reasons as stated in the rejection of claim 34.

14. As to claim 52, it is rejected for the same reasons as stated in the rejection of claim 35.

15. Claims 36 and 38-42 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nelson (US 2001/0049686) in view of Poole (US 6,314,445 B1), and further in view of Delagi et al. (hereinafter Delagi) (US 3,858,182).

16. As to claim 36, Nelson fails to explicitly teach the system of claim 30, wherein the routines in the threads interface layer maintain a set of context information for each thread in terms of the virtual machine. However, Delagi teaches saving context information from a previous virtual machine to a current virtual machine *(context information, virtual machine, restore previous context, col 8 lines 6-18)*. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the Delagi feature of maintaining context information in terms of the virtual machine to the existing system so that the virtual machine can resume the execution of any program when needed, therefore, increasing efficiency and reducing overhad involved with resuming execution (col 8, lines 6-15).

17. As to claim 38-42, it is rejected for the same reasons as stated in the rejection of claim 36. In addition, "Official Notice" is taken that both the concept and advantages of providing that "native threads support routines include a routine for resuming, waiting for completion, yielding execution, stopping execution and cleaning up of a particular thread" is well known and expected in the art. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include these basic and standard functions/routines to the existing system for the reason of improving the control of thread/task management scheduling.

18. **Claims 37 is rejected under 35 U.S.C. 103(a) as being unpatentable over Nelson (US 2001/0049686) in view of Poole (US 6,314,445 B1), and further in view of Bucher (US 5,421,014).**

19. As to claim 37, Nelson teaches wherein the native threads interface layer is adapted to an operating system of the underlying platform (*Java Native Interface ("JNI") Layer 604 includes a library of C and/or C++ methods configured to define a Java Virtual Machine ("JVM") that provides translation of CMIS to Java.", "this layer also includes Solaris threads configured to provide additional support when the MPA is running in conjunction with the Solaris operating system", [0039]*) but fails to explicitly teach having a set of context information for each thread. However, Bucher shows that it is common knowledge in the art of context switching that the multi-threading and context switching for threads that context information for multiple are used

to uniquely identify threads and the information (context information) that is necessary to relate to each of the threads *("Multi-threading is accomplished by temporarily suspending execution of one thread and beginning execution of another, eventually restoring the suspended thread to complete its execution. In order to perform multi-threaded operations, the computer is required to internally store context information for multiple threads, update context information at appropriate times during execution of the thread, and uniquely identify threads and the context information related to each of the threads.", col. 1, lines 59-68)*. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of having context information for each thread for the reason of improving context switching and multi-threading by having the necessary information which uniquely identifies the particular thread *(col. 1, lines 59-68)*.

20. **Claims 43-46 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nelson (US 2001/0049686) in view of Poole (US 6,314,445 B1), and further in view of Farrell et al. (hereinafter Farrell) (US 5,630,128).**

21. As to claims 43-46, Nelson teaches having native threads support routines but fails to explicitly teach the system of claim 30, wherein they include a routine for setting a priority, obtaining a priority, obtaining an identifier, and finally executing of a particular thread. However, Farrell discloses threads being set a priority. The priority is also being obtained and classified. The threads are being identified and selected before they are finally executed *("The* *invention resides in a multitasking operating system which permits application programs (and their developers) to influence a schedule of execution of program threads derived from the application programs, by specifying parameters for the program threads. The parameters indicate each thread's priority level and dispatch class in which the thread resides. Based on these parameters, the operating system schedules the program threads for execution in the following fashion. The operating system selects the highest priority program thread which is available for execution from each dispatch class for execution by a processor. According to one feature of the invention, an application program thread can change the dispatch class in which another program thread resides. According to another feature of the invention, an executing program thread can voluntarily yield to a specified program thread in the same dispatch class or permit the highest priority available thread in the dispatch class to be queued on the run list with itself being available and in contention for the run list. Both of these features permit application programs to influence the schedule of execution of the program threads.", col. 2, lines 24-45).* It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of executing a thread based on priority for the reason of increasing the control of the scheduling of the program with priority scheduling. A multitasking operation system permits application programs to influence a schedule of execution of program threads *(see Abstract).*

Response to Arguments

22. Applicant argues on page 7 that Nelson teaches only threads support that <u>*does*</u> *depend on an underlying platform. Nelson clearly states in the description the Java virtual machine of the JNI layer 604 that this layer also includes Solaris threads configured to provide additional* *support when the MPA is running in conjunction with the Solaris operating system (available commercially from Sun Microsystems...*

In response, Examiner respectfully disagrees. First, the above assertion is only one embodiment of Nelson [0039] and Nelson says that the layer can also (not must only) include Solaris threads. Nelson also teaches that the combination of Java programming code and the Java Virtual Machine allows the creation of platform-independent software *([0006] "The Java programming language has emerged as a tool for producing software efficiently. In particular, the combination of Java programming code and the Java Virtual Machine ("JVM") allows the creation of platform-independent software that can be installed dynamically and/or remotely. Thus, the use of Java-coded network software applications, agents, and other entities can alleviate the above-described expenses and difficulties associated with producing network management software using traditional languages and methods. In addition, the Java programming language is richer than the languages used currently to provide network management software. Thus, Java-based software management software application, agents, and entities having more functionality can be developed.").*

23. Applicant argues on page 8 that the presented prior art does not teach or suggest a system having a two-tier arrangement for adapting threads support in a virtual machine to an underlying platform.

In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., "two-tier arrangement) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Conclusion

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Kenneth Tang whose telephone number is (703) 305-5334. The examiner can normally be reached on 8:30AM - 7:00PM, Monday through Thursday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (703) 305-9678. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 746-7140.

[illegible]

Kt
1/10/04

MENG-AL T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100